

1er Recuperatorio

Algoritmos y Estructuras de Datos 3 – DC, FCEyN, UBA

09/12/2020

Para realizar consultas, deben conectarse al aula de Zoom habitual (<https://zoom.us/my/dc.aula03>) y avisarle al *host* por Discord que desean consultar. Serán agregados a una lista de espera visible en <https://docs.google.com/spreadsheets/d/1R80e2WI5nI1lM7TfZKZfwF4eLQrwJyb4eWMAq0PBg0/edit?usp=sharing> y, cuando llegue su turno, transferidos a una *breakout room* con algún docente al que realizarle su consulta.

Las aclaraciones de enunciado que podamos llegar a hacer van a ser comunicadas por Zoom y por Discord.

El examen transcurre de 17:00 a 21:00 hs. De 21:00 a 21:15 no contestaremos más consultas de enunciado y **sólo** estaremos en la sala de Zoom para resolver cuestiones relacionadas a la entrega por Campus (<https://campus.exactas.uba.ar/mod/assign/view.php?id=151447>). A las 21:15 se cerrará la sesión de Zoom y tendrán hasta las 21:45 para realizar la entrega. Tengan en cuenta que desde las 21:15 los docentes ya no asistiremos con cuestiones de entrega.

Sólo en caso de que el Campus estuviera saturado y no funcionara, sería adecuado realizar la entrega por mail a algo3-doc@dc.uba.ar. Sólo en caso de entregar por Campus, el archivo podrá sobreescribirse una cantidad ilimitada de veces hasta la hora de entrega. En caso de entregar por Campus, independientemente de si sobreescriben o no, deberán confirmar su entrega definitiva (que ya no podrá sobreescribirse).

El examen **debe** realizarse a mano. **Deben** numerar sus hojas y escribir en ellas sus nombres. Al finalizar, deben escanearlo o fotografiarlo, preferentemente con CamScanner o similar. El resultado **debe** ser un documento legible (buena iluminación, buena resolución, buena orientación, no fotos cortadas, etc.), **¡verificarlo!**

Deben subir o bien un único archivo PDF o bien un único archivo comprimido que contenga las imágenes. En cualquier caso, **debe** estar nombrado **apellido_nombre.extensión** y **debe** haber un orden de lectura claro (pueden nombrar los archivos de las imágenes de acuerdo al orden de lectura).

El examen es personal y pueden usar las teóricas, las clases prácticas y las guías de ejercicios, citando claramente. Las respuestas deben estar debidamente justificadas.

El examen se aprueba con al menos 2 ejercicios bien.

- 1) Un vértice v de un grafo G es un punto de articulación si $G - v$ tiene más componentes conexas que G . Por otro lado, un grafo se dice biconexo si es conexo y no tiene puntos de articulación.

- a) Demostrar que todo grafo de n vértices que tenga al menos

$$\frac{(n-1)(n-2)}{2} + 2$$

aristas es biconexo.

- b) ¿Se puede dar una cota mejor que funcione a partir de algún n_0 ? Es decir, ¿existe

$$c(n) < \frac{(n-1)(n-2)}{2} + 2$$

tal que todo grafo de $n \geq n_0$ vértices que tenga al menos $c(n)$ aristas sea biconexo?

- 2) Un problema común en los procesadores de texto es decidir dónde particionar un *string* S que representa un párrafo para formar los distintos reglones. En este problema, el objetivo es maximizar la belleza visual del texto resultante. Para ello, se define una matriz de belleza B con $|S| \times |S|$ entradas naturales, donde $B[i, j]$ representa la belleza que tendría un reglón que empieza en $S[i+1]$

y termina en $S[j]$ ($B[i, j] = 0$ si $j \leq i$). Luego, el problema de particionado en reglones consiste en determinar una secuencia de puntos de partición $0 = p_0 < p_1 < \dots < p_{k+1} = |S|$ que maximice $\sum_{i=0}^k B[p_i, p_{i+1}]$. **Ayuda:** notar que para partir S en k reglones tenemos que elegir k de las $n - 1$ posibles puntos de partición. Luego, existen $\sum_{k=0}^{n-1} \binom{n-1}{k}$ formas de partir un párrafo.

- a) Definir la función $belleza_S : \mathbb{N} \rightarrow \mathbb{N}$ tal que $belleza_S(j)$ es la máxima belleza posible para el *substring* de S que empieza en $S[1]$ y termina en $S[j]$.
 - b) Mostrar que $belleza_S$ tiene la propiedad de superposición de subproblemas.
 - c) Definir un algoritmo *top-down* para calcular la máxima belleza posible (no hace falta calcular los puntos de partición) cuando S y B son dados como input, indicando claramente las estructuras de datos utilizadas y la complejidad resultante. Nota: para que el ejercicio se considere bien es necesario que la complejidad del algoritmo sea $\mathcal{O}(|S|^2)$.
- 3) En la ciudad hay k espías y cada uno se oculta en una esquina distinta. Es necesario elegir una esquina para que todos los espías se reúnan y puedan intercambiar información secreta. Una vez elegida la esquina, la misma será comunicada simultáneamente a todos los espías, quienes se dirigirán hacia ella de inmediato. La reunión comenzará en cuanto todos los espías hayan llegado a la esquina elegida.
- La ciudad está formada por n esquinas y m calles. Cada esquina se identifica por un entero distinto entre 1 y n . Cada calle conecta determinado par de esquinas y se puede recorrer en ambos sentidos. Se puede ir de cualquier esquina a cualquier otra recorriendo las calles, pasando eventualmente por esquinas intermedias. Cada espía tarda exactamente 42 segundos en recorrer cada calle.
- Diseñar un algoritmo eficiente (no simplemente polinomial) para determinar cuáles esquinas permiten realizar la reunión lo antes posible. La entrada del algoritmo es la cantidad n de esquinas, la cantidad m de calles, la cantidad k de espías, para cada calle las esquinas que conecta, y para cada espía la esquina en la que está oculto. Indicar claramente las estructuras de datos utilizadas, mostrar que el algoritmo propuesto es correcto y explicar su complejidad. Justificar claramente, citando las afirmaciones del enunciado cuando corresponda. Para que el ejercicio se considere bien es necesario que la complejidad del algoritmo sea $\mathcal{O}(km)$.
- 4) El algoritmo de Kruskal con orden de selección es una variante del algoritmo de Kruskal donde a cada arista e se le asigna una prioridad $q(e)$ además de su peso $p(e)$. Luego, si en alguna iteración del algoritmo de Kruskal hay más de una arista posible para ser agregada, entre esas opciones se elige alguna de mínima prioridad. Demostrar que para todo árbol generador mínimo T de G , si las prioridades de asignación están definidas por la función

$$q_T(e) = \begin{cases} 0 & \text{si } e \in T \\ 1 & \text{si } e \notin T \end{cases}$$

entonces se obtiene T como resultado del algoritmo de Kruskal con orden de selección ejecutado sobre G .